REMARKS

Claims 1-7 and 9-12 are pending in the application. Claims 1 and 7 have been amended. Reconsideration of this application is respectfully requested.

The Office Action rejects claims 1-7, 9, 10 and 12 under 35 U.S.C 103(a) as unpatentable over U.S. Patent Publication No. 2004/0133444 to Defaix et al., hereafter Defaix in view of U.S. Patent No. 6,449,624 to Hammack et al., hereafter Hammack.

This rejection is respectfully traversed. Independent claims 1 and 7 have been amended to recite:

"a validation function operable on said processor to determine whether said import object is eligible for automatic check-in, wherein said validation function comprises:

determining if said import object already exists as an existing object in said process control system;

if said import object already exists as said existing object, determining if said existing object has a status of checked-in;

if said existing object is checked in, determining if said user has permission to check-in; and

if said user has permission to check-in said existing object, then automatically locking said status of said existing object so as to validate said import object".

Independent claims 1 and 7 recite a validation function that comprises a sequence of four steps in which each step is conditioned on the preceding step. Independent claims 1 and 7 have been amended to change "locking" to "automatically locking" in the third step of the sequence. Support for this amendment is step 140 of drawing Fig. 1B and paragraph 0026 of the specification.

The Examiner contends that Defaix discloses the validation function, citing paragraphs 0035, 0032, 0007, 0039, 0040 and 0029. These paragraphs do not disclose the claimed sequence of four steps. Only paragraph 0035 describes a validation function that is used by server 100 to validate a request by a user of client 300 to check-in a file, which the user has modified. The validation function consists solely of determining whether the file is locked. If so, the user's request is denied. If not, "then the central server 100 informs the client 300 through proxy server 200 that its update is allowed". This validation function of Defaix has no relevance to any of the four steps of the validation function recited in independent claims 1 and 7. In fact, paragraph 0035 does not mention any of the four steps of the validation function recited in independent claims 1 and 7.

With respect to the first of the four steps, the Examiner cites paragraphs 0032 and 0035. As noted above, paragraph 0035 merely describes checking to see if the file that is the subject of the request is locked out. This is not a determination of whether the import object already exists in the process control system as claimed. Since the files that client 300 uses can only be files stored in repository 102 (paragraph 0032), the server has no need to make this determination. Therefore, paragraphs 0032 and 0035 do not support the Examiner's position. Thus, Defaix lacks the first step of the claimed validation function.

In paragraph 18 of the Office Action, the Examiner contends Defaix "not only teaches using files stored in the repository, but also teaches files that are not initially stored in the repository, i.e., 'some files will also already be stored in the sandbox', see 0032". This contention is mistaken. The client's sandbox contains only copies of versions of files stored in the repository. Paragraph 0032 clearly states that the sandbox 306 stores "local working copies of files from a corresponding project on the central server 100" and the "files in sandbox 306 are (possibly modified) particular versions of files from the repository 102". Paragraph 0032 further notes that the client "preferably does not have a local bulk data cache for the file contents" but can obtain them quickly via proxy server 200 as necessary". The statement that "some files will also already be

stored in the sandbox 306" refers to the working copies set forth earlier in paragraph 0032. That is, some of the working copies of files in the repository may not contain the file contents, but the client can quickly obtain the file contents from proxy server 200. Therefore, the Examiner's contention is mistaken.

The Examiner concludes, based on the mistaken contention, that it would have been obvious that "developers are allowed to add new files to the repository over time". There is no teaching or disclosure in Defaix that describes this feature. The Examiner further argues, based on the mistaken contention, that "the operation of checking whether a file is locked performed with a request to check in a file (0035) would necessarily check whether a file exists first since the locked status of a file cannot be determined for a file that does not yet exist in the repository". This argument is erroneous not only because it is based on a mistaken contention, but also because Defaix makes no teaching or disclosure of how the files of a new project are loaded into the repository. For the foregoing reasons, the Examiner's contentions and arguments with respect to the first step are without merit. Therefore, Defaix lacks the first step.

With respect to the second step of the validation function, the Examiner cites paragraphs 0007 and 0035. Paragraph 0007 located in the Background of the Invention section, describes a prior art locked *checkout* mechanism to control access to files. Defaix does not use this mechanism, but uses the mechanism described in paragraph 0038 that allows a first one of many clients working on a project to request a lock on a file that the first client is working on. Should the lock be granted, a second client working on the same file will be prevented from checking in a modified version of the file (paragraph 0035). In contrast, the second step recites "if said import object already exists as said existing object, determining if said existing object has a status of *checked-in*". Since all of the existing files of a project are already in the repository, there is no need for Defaix to respond to a request to check in a modified file to determine if the existing version of the modified file is checked in. Thus, Defaix merely discloses checking in of modified versions and in so doing merely determines whether the existing file in the repository is locked by another client. Thus, in Defaix's system there is no

need to determine if the existing file has a status of checked in. Therefore, Defaix lacks the second step of the claimed validation function.

In paragraph 19 of the Office Action, the Examiner argues "that checking if a file is locked is the same as checking if a file is checked in since a file is locked when a file is checked out (see 0007). Therefore, if a file is locked, then it is not checked in, and if a file is not locked, then it is checked in". This argument is mistaken. All of the existing files in repository 102 have a status of checked in. Defaix does not disclose or teach a checkout system that performs a lock at the time of checkout. Rather, Defaix distinguishes from the prior art system of paragraph 0007, by allowing multiple clients working on a project to access and possess working copies of the existing files in repository 102 that pertain to the project. So, in Defaix checkout is a meaningless event unless the existing file is locked. In order to lock an existing file, a client must request a lock. The procedure for granting the lock is shown in Fig. 5 and described in paragraph 0038. Notice that the fact that an existing file is locked by a first client prevents a second client from checking in a modified version of the existing file, but does not prevent the second client or another client from obtaining a copy of the existing file for its sandbox. For these reasons, there is no need for Defaix to determine "if said existing object has a status of checked in" because all of the existing files are checked in. Therefore, Defaix lacks the second step.

With respect to the third step of the validation function, the Examiner cites paragraphs 0039 and 0040. However, it is questionable that the password procedure described in paragraphs 0039 and 0040 is used by the validation function described in paragraph 0035. Paragraphs 0039 and 0040 describe a password procedure for proxy 200 and client 300 that is used by server 100 to grant permissions of proxy 200 and client 300 and to add the permissions to an access control list 110, which is then used to validate requests by proxy 200 and client 300. Defaix discloses in Fig. 7 and paragraph 0043 that the proxy server in processing the client's request for access to a file uses a step 506 to determine if the user has permission to access and receive a particular file, but otherwise does not disclose a permission step except for the case of checking in a modified version

of a locked existing file (paragraph 0035). However, there is no description that once this request is granted, the control list would later be consulted for a request concerning the file already checked out to client 300. Thus, paragraph 0035 when considered with paragraphs 0039 and 0040 does not explicitly teach that server 100 uses the password procedure in the validation function described in paragraph 0035. Therefore, Defaix lacks the third step of the claimed validation function.

In paragraph 20 of the Office Action, the Examiner submits that the access control list is used for check in of a modified version because it is used to control who has access to the existing files in repository 102. However, there is no teaching or disclosure in Defaix that the access control list is used at the time of check in of a modified version of an existing file. Therefore, the Examiner's argument is without merit. Thus, Defaix lacks the third step.

With respect to the fourth step of the claimed validation function, the Examiner admits that Defaix does not disclose this step. The Examiner contends that Defaix's teaching is "such that only the developer who owns the lock can modify the file by checking in a new version (see 0029)." The Examiner concludes that it would be obvious "to lock the status of the existing object to prevent other developers from modifying the file when the file is being checked in". This contention and conclusion is erroneous. The prior art locked checkout mechanism disclosed by Defaix in paragraph (0007) locks an object *upon checkout* and not *upon check-in* as recited in claimed fourth step of the validation function. There is no teaching in Defaix to modify the locked *checkout* mechanism of paragraph 0007 to be a locked *check-in* mechanism or to even use the paragraph 0007 procedure. Therefore, the Examiner's contention and conclusion are erroneous.

In paragraph 21 of the Office Action, the Examiner argues that in "the situation where a check in is performed on a file that is not locked, it would have been obvious to lock the file while the check in is in progress to prevent other developers from modifying the file when the file is being checked in". However, there is no teaching in Defaix for

this conclusion of obviousness. In fact, the only teaching in Defaix for a lock procedure is shown in Fig. 5 and described in paragraph 0038 for locking a file. This procedure is independent of check in or checkout and must be separately requested by the client.

Amended independent claims 1 and 7 recite:

"if said user has permission to check-in said existing object, then automatically locking said status of said existing object so as to validate said import object".

Defaix does not automatically lock the "status of said existing object so as to validate said import object". Therefore, Defaix lacks the fourth step.

Hammack, which was cited for a different reason, does not supply the above mentioned deficiencies of Defaix. Therefore, claims 1-7, 9, 10 and 12 are unobvious in view of the combination of Defaix and Hammack.

For the reasons set forth above, it is submitted that the rejection of claims 1-7, 9, 10 and 12 under 35 U.S.C. 103(a) is obviated by the amendment and should be withdrawn.

The Office Action rejects claim 11 under 35 U.S.C 103(a) as unpatentable over Defaix in view Hammack as applied to claim 10, and further in view of U.S. Patent Publication No. 2002/0019827 to Shiman et al., hereafter Shiman.

This rejection is obviated by the amendment to independent claim 7 for the reason that the combination of Defaix and Hammack lacks the above noted deficiencies of independent claim 7, from which claim 11 depends via intervening claim 10. Shiman, which was cited for a different reason, does not supply the deficiencies. Therefore, claim 11 is unobvious over the combination of Defaix and Hammack and Shiman.

For the reasons set forth above, it is submitted that the rejection of claim 11 under 35 U.S.C. 103(a) is obviated by the amendment and should be withdrawn.

It is respectfully requested for the reasons set forth above that the rejections under 35 U.S.C. 103(a) be withdrawn, that claims 1-7 and 9-12 be allowed and that this application be passed to issue.

For the reasons set forth above, it is submitted that this amendment places the application in condition for allowance. Accordingly, it is respectfully requested that this application be allowed and passed to issue. If this amendment is deemed to not place the application in condition for allowance, it is respectfully requested that it be entered for the purpose of appeal.

Respectfully Submitted,

Date: 6/12/09

Paul D. Greeley
Reg. No. 31,019
Attorney for Applicant
Ohlandt, Greeley, Ruggiero & Perle, L.L.P.
One Landmark Square, 10th Floor
Stamford, CT 06901-2682
(203) 327-4500